Titel

[Abkürzungsverzeichnis 3](#_Toc102660220)

[Modbus 4](#_Toc102660221)

[Verzeichnisse 6](#_Toc102660222)

[Quellenverzeichnis 6](#_Toc102660223)

[Abbildungsverzeichnis 7](#_Toc102660224)

[Tabellenverzeichnis 8](#_Toc102660225)

## Abkürzungsverzeichnis

PLC: Programmable Logic Controller

TCP: Transmission Control Protocol

IP: Internet Protocol

TCP/IP: Transmission Control Protocol/Internet Protocol (Internet Protocol Suite)

RTU: Remote Terminal Unit

ASCII: American Standard Code For Information Interchange

ADU: Application Data Unit

PDU: Protocol Data Unit

Dec.: Dezimal

Hex.: Hexadezimal

Bin.: Binär

### Modbus

Modbus ist eine Datenübertragungsprotokoll, welches von Modicon, jetzt Schneider Electric, im Jahr 1979 entwickelt wurde. Es wurde schon von hunderten verschiedenen Herstellern bei tausenden verschiedenen Geräten eingesetzt und ist damit der de facto Standard um PLCs, Computer, Sensoren und Aktoren miteinander zu verbinden [1, S. 508], [2]. Es gibt verschiedene Versionen von Modbus, unter anderem Modbus RTU, Modbus ASCII, Modbus über TCP/IP, Modbus Plus und Modbus Enron [1, S. 509], [2]–[4]. Serieller Modbus (Modbus RTU, Modbus ASCII,) beruht auf dem Master/Slave-System, wo eine Master-System bis zu 247 Slave-Systeme nach Daten abfragen kann [1, S. 508]. Die folgenden Kapitel beinhalten nur Modbus RTU.

### Modbus Frame

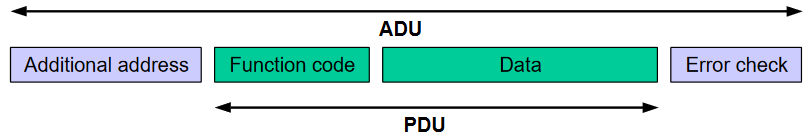
Der Master sendet Nachrichten, auch Queries genannt an jeweils einen, durch seine Adresse spezifizierten, Slave. Dieser antwortet dem Master mit einer Response. Eine vollständige Nachricht im Modbus-Protokoll ist in Abb. 1 zu sehen, wird Application Data Unit (ADU) genannt und besteht aus einer Adresse, der Protocol Data Unit (PDU), bestehend aus dem Funktionscode und den zu übertragenden Daten, und schlussendlich dem Fehlercheck.   


Abb. 1 Komponenten Modbus Nachricht [5, S. 3]

Die Query sagt dem adressierten Slave via dem Funktionscode was er tun soll. Beispielsweise sagt der Code 03, dass der Slave Register auslesen und mit deren Inhalt antworten soll. Im Datenfeld ist dann das Register, welches zuerst ausgelesen werden soll, und die Anzahl der drauffolgenden auszulesenden Register. Der Slave antwortet darauf mit seiner Adresse und dem Funktionscode, welchen er ausgeführt hat. Im Datenfeld sind dann aber die ausgelesenen Registerdaten enthalten [6, S. 14]. Die Modbus Nachricht, auch Frame genannt, startet und endet mit mindestens 3,5 Bitübertragungszeitdauer Stille. Wenn in dem Frame mehr als 1,5 Bitübertragungszeitdauer Stille enthalten sind, wird es vom Empfänger verworfen. Das Adressfeld enthält acht Bit und es gibt 247 valide Adressen. Der Master adressiert den Slave mit seiner Adresse. Ein Slave antwortet dem Master mit seiner eigenen Adresse im Adressfeld. Im Funktionscodefeld steht eine acht Bit lange Nachricht, die dem Slave sagt, was er genau zu machen hat. Es gibt elf verschiedene Funktionscodes, welche alle Modbusübertragungsmedien unterstützen, welche in Tabelle I dargestellt sind und mehrere Dutzende, welche Übertragungsmedium spezifisch sind. Das Datenfeld ist n\*8 Bit lang und enthält die zu übertragenden Daten. Als letztes kommt das 16-bit-lange Fehlercheckfeld, wo mit Hilfe von Cyclical-Redundancy-Check-Rechnungen die empfangene Nachricht überprüft wird [6, S. 18–25].

TABELLE ALLGEMEINE MODBUS FUNKTIONEN [6, S. 31–66]

|  |  |  |
| --- | --- | --- |
| Code in Dec. | Name | Funktion |
| 01 | Read Coil Status | Liest den AN/AUS Status eines Coils |
| 02 | Read Input Status | Liest den AN/AUS Status eines Inputs |
| 03 | Read Holding Register | Liest die Werte von Holding Registern |
| 04 | Read Input Register | Liest die Werte von Input Registern |
| 05 | Force Single Coil | Setzt den AN/AUS Status eines Coils |
| 06 | Preset Single Register | Setzt die Werte von einem Holding Register |
| 07 | Read Exception Status | Liest den Status von prädefinierten Status Coils. Bspw.: Maschine in Betrieb |
| 08 | Diagnostic | Mehrere Unterfunktionen für verschieden Status von Slaves |
| 15 | Force Multiple Coils | Setzt den AN/AUS Status mehrere Coils |
| 16 | Preset Multiple Register | Setzt die Werte von mehreren Holding Register |
| 17 | Report Slave ID | Zeigt die Slave ID an |



Abb. Beispiel Modbus Funktion 03[6, S. 15]

In Abb. 2 kann man eine Beispielsregisterabfrage sehen. Es wurde nur die PDU dargestellt, bei der gesamten Nachricht wurde am Anfang noch die Adresse und am Ende die CRC-Summe übermittelt. Alle Registeradressen und Registerwerte werden in einen High- und Lowteil aufgeteilt, welche hintereinander gesetzt werden müssen. Register Value Hi \* 0x100 + Register Value Lo = Register Value (1). Der Master fragt dort die Daten von Register 0x6B, 0x6C und 0x6D, indem das Startregister und die Anzahl der darauffolgenden Register definiert werden. Der Slave antwortet daraufhin erst mit der Funktion und den Bitanzahl der Antwortwerte. Danach mir den Werten der abgefragten Register. So weiß der Master, dass Register 0x6B = 0x022B, Register 0x6C = 0x000 und Register 0x6D = 0x0064 ist.

# Verzeichnisse

## Quellenverzeichnis

[1] B. Drury, *The control techniques drives and controls handbook*, 2nd ed. Stevenage: Institution of Engineering and Technology, 2009.

[2] „Modbus FAQ“. https://modbus.org/faq.php (zugegriffen 5. Mai 2022).

[3] „About Modbus | Simply Modbus Software“. http://www.simplymodbus.ca/FAQ.htm (zugegriffen 5. Mai 2022).

[4] „Modbus Plus | Schneider Electric USA“. https://www.se.com/us/en/product-range/576-modbus-plus/ (zugegriffen 5. Mai 2022).

[5] Modbus Organization, Inc, „MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b3“. 26. April 2012. [Online]. Verfügbar unter: https://modbus.org/specs.php

[6] MODICON, Inc., Industrial Automation Systems, „Modicon Modbus Protocol Reference Guide“. Juni 1996.

## Abbildungsverzeichnis

[Abb. 1 OSI-MODELL 6](file:///D:\STuff\Uni\6.%20Semester\TITEL4%20-%20Kopie.docx#_Toc94458933)

[Abb. 2 COMMUNICATION LAYERS [6, S. 18] 7](#_Toc94458934)

[Abb. 3 MAC FRAME[13, S. 108–113] 8](#_Toc94458935)

[Abb. 4 KABELSCHNÜFFLER ZABBIX DIDACTUM 12](#_Toc94458936)

[Abb. 5 ZABBIX DASHBOARD 14](#_Toc94458937)

[Abb. 6 ZABBIX HOSTS 15](#_Toc94458938)

[Abb. 7 ZABBIX ITEM 16](#_Toc94458939)

[Abb. 8 ZABBIX TRIGGER 18](#_Toc94458940)

[Abb. 9 ZABBIX DASHBOARD 18](#_Toc94458941)

[Abb. 10 ZABBIX WEBHOOK DISCORD 19](#_Toc94458942)

[Abb. 11 ZABBIX USER 20](#_Toc94458943)

[Abb. 12 ZABBIX LOW-LEVEL DISCOVERY RULE 20](#_Toc94458944)

[Abb. 13 GLEICHER AGENT ACTIVE UND PASSIV 21](#_Toc94458945)

[Abb. 14 DIDACTUM MONITORING SYSTEM 500 II [25] 22](#_Toc94458946)

[Abb. 15 DIDACTUM TEMPERATUE SENSOR [27] 22](#_Toc94458947)

[Abb. 16 CONTAINER [28] 23](#_Toc94458948)

[Abb. 17 DOCKER CONTAINER 23](#_Toc94458949)

[Abb. 18 TEMPERATURVERLAUF EINES SENSORS IN ZABBIX 24](#_Toc94458950)

[Abb. 19 ALARM UND ENTWARNUNG IM SOFORTNACHRICHTENDIENST 24](#_Toc94458951)

[Abb. 20 ZABBIX DOCKER DATEIEN 25](#_Toc94458952)

[Abb. 21 ZABBIX IN DOCKER 25](#_Toc94458953)

[Abb. 22 DOCKER ZABBIX MIT AGENT INSTALLATION 26](#_Toc94458954)

[Abb. 23 DIDACTUM EINSTELLUNGEN 27](#_Toc94458955)

[Abb. 24 DIDACTUM SYSTEMBAUM 27](#_Toc94458956)

[Abb. 25 VERLINKTES TEMPLATE 28](#_Toc94458957)

[Abb. 26 ZABBIX AUTOREGISTRATION ACTION 29](#_Toc94458958)

[Abb. 27 ZABBIX DISCOVERY RULE 30](#_Toc94458959)

[Abb. 28 ZABBIX TRIGGER 31](#_Toc94458960)

[Abb. 29 ZABBIX NACHRICHT TEMPLATE 32](#_Toc94458961)

[Abb. 30 DISCORD WARNUNG 32](#_Toc94458962)

[Abb. 31 TEMPERATUR- UND FEUCHTIGKEITSBEREICH [32] 33](#_Toc94458963)

[Abb. 32 ZABBIX MAP BSP.1 [34] 36](#_Toc94458964)

[Abb. 33 ZABBIX MAP BSP.2 [35] 37](#_Toc94458965)

## Tabellenverzeichnis

[TABELLE I ARP [21] 12](#_Toc94458966)

[TABELLE II SNMP [21] 13](#_Toc94458967)

[TABELLE III ZABBIX ITEMS [24, Teil 7.2.3] 17](#_Toc94458968)